CvH Proteomics — Normalization Pipeline

Cancer Survivor (CR) vs. Healthy (PPS) Skeletal Muscle DIA-MS Proteomics

DTL

2026-02-15

Table of contents

0 — Setup

This document implements a normalization pipeline for the CvH DIA-MS proteomics dataset using the `proteoDA` package. The pipeline follows current best practices for label-free quantitative proteomics preprocessing, drawing on recommendations from Välikangas et al. (2018) and the `proteoDA` workflow.

Study overview: Cancer survivors (CR) who completed 12 weeks of resistance training were compared with healthy age-matched controls (PPS). CR participants were randomized to creatine (CRE) or placebo (PLA) supplementation. Vastus lateralis biopsies were collected at baseline (T1) and post-training (T2) for CR participants, and at a single timepoint (T1) for healthy controls.

Load required packages and set up paths.

```r
# Package management
if (!requireNamespace("pacman", quietly = TRUE)) install.packages("pacman")
pacman::p_load(
  proteoDA,
  readxl, readr, dplyr, tidyr, stringr,
  ggplot2, patchwork, ggrepel, knitr, here
)

# Paths
base_dir   <- here::here()
input_dir  <- file.path(base_dir, "00_input")
report_dir <- file.path(base_dir, "01_normalization", "b_reports")
data_dir   <- file.path(base_dir, "01_normalization", "c_data")

dir.create(report_dir, recursive = TRUE, showWarnings = FALSE)
dir.create(data_dir,   recursive = TRUE, showWarnings = FALSE)

# Color palette - consistent across all CvH scripts
pal_group <- c(CR_CRE = "#2166AC", CR_PLA = "#67A9CF", PPS = "#B2182B")
pal_group_time <- c(
  CR_T1 = "#92C5DE", CR_T2 = "#2166AC",
  H_T1  = "#B2182B"
)
pal_supplement <- c(CRE = "#2166AC", PLA = "#67A9CF", `NA` = "#999999")
shape_tp <- c(T1 = 16, T2 = 17)  # circle, triangle
```

1 — Load & Validate Data

We begin by reading the raw DIA-MS intensity data and the sample metadata. The raw file contains protein-level intensities exported from the search engine, with annotation columns (UniProt ID, gene symbol, description) followed by one column per sample.

Separate annotation columns from sample intensities for the DAList structure. The annotation/intensity split is required because `proteoDA` stores them in different slots.

```r
# Raw intensity data
raw <- read_excel(file.path(input_dir, "CvH_raw.xlsx"))

# Separate annotation from intensity
annot_cols <- c("uniprot_id", "protein", "gene", "description", "n_seq")
annotation <- raw[, annot_cols]
intensity  <- raw[, setdiff(names(raw), annot_cols)]

cat(sprintf("Raw data: %d proteins x %d samples\n", nrow(raw), ncol(intensity)))
```

```
Raw data: 3172 proteins x 41 samples
```

```r
# Metadata
metadata <- read.csv(file.path(input_dir, "CvH_meta.csv"), stringsAsFactors = FALSE)
rownames(metadata) <- metadata$Col_ID

cat(sprintf("Metadata: %d samples\n", nrow(metadata)))
```

```
Metadata: 41 samples
```

A mismatch between sample identifiers in the intensity matrix and metadata would silently corrupt everything downstream, so we halt immediately if the sets differ.

```r
# Critical check: sample columns must match metadata rows
data_samples <- colnames(intensity)
meta_samples <- metadata$Col_ID

stopifnot(
  "Sample mismatch between data and metadata" =
    setequal(data_samples, meta_samples)
)
```

```
# Reorder intensity columns to match metadata row order
intensity <- intensity[, meta_samples]

cat("Validation passed: all", length(meta_samples), "samples aligned.\n")
```

```
Validation passed: all 41 samples aligned.
```

Verify the expected experimental design before proceeding.

```
metadata %>%
  count(Group, Timepoint, Group_Time, Supplement) %>%
  kable(caption = "Sample distribution across experimental groups")
```

Table 1: Sample distribution across experimental groups

Group	Timepoint	Group_Time	Supplement	n
CR_CRE	T1	CRE_T1	CRE	9
CR_CRE	T2	CRE_T2	CRE	8
CR_PLA	T1	PLA_T1	PLA	8
CR_PLA	T2	PLA_T2	PLA	6
PPS	T1	H_T1	NA	10

2 — Assemble DAList

The `proteoDA` package uses a `DAList` to hold intensity data, protein annotation, and sample metadata together, ensuring consistent indexing throughout. Before assembly, we annotate proteins against the Human Protein Atlas (HPA) skeletal muscle gene list (Uhlen et al., 2015) to flag tissue-relevant proteins for the filtering step.

When multiple rows share the same UniProt accession (e.g., isoforms collapsed by the search engine), we retain the entry with the highest mean intensity to ensure a unique row key.

```
# HPA skeletal muscle annotation
hpa_file <- file.path(input_dir, "HPA_skeletal_muscle_annotations.tsv")
stopifnot("HPA annotations file not found" = file.exists(hpa_file))
hpa_genes <- unique(read_tsv(hpa_file, show_col_types = FALSE)$Gene)

annotation <- annotation %>%
  mutate(in_hpa_muscle = gene %in% hpa_genes)
```

```r
cat(sprintf("HPA muscle genes: %d | Matched in dataset: %d / %d\n",
            length(hpa_genes), sum(annotation$in_hpa_muscle), nrow(annotation)))
```

```
HPA muscle genes: 12887 | Matched in dataset: 2962 / 3172
```

```r
# Check for duplicate uniprot_ids
dup_ids <- annotation$uniprot_id[duplicated(annotation$uniprot_id)]

if (length(dup_ids) > 0) {
  cat(sprintf("Found %d duplicate uniprot_ids - deduplicating by highest mean intensity.\n",
              length(dup_ids)))

  # Calculate row means for deduplication
  intensity_num <- as.data.frame(lapply(intensity, as.numeric))
  annotation$row_mean <- rowMeans(intensity_num, na.rm = TRUE)

  keep_idx <- annotation %>%
    mutate(row_idx = row_number()) %>%
    group_by(uniprot_id) %>%
    slice_max(row_mean, n = 1, with_ties = FALSE) %>%
    pull(row_idx)

  annotation <- annotation[keep_idx, ]
  intensity  <- intensity[keep_idx, ]
  annotation$row_mean <- NULL

  cat(sprintf("After deduplication: %d unique proteins\n", nrow(annotation)))
} else {
  cat("No duplicate uniprot_ids found.\n")
}
```

```
No duplicate uniprot_ids found.
```

Bundle all three components so `proteoDA` maintains consistent alignment throughout (Herbrich et al., 2013).

```r
# Convert intensity to numeric matrix
intensity_mat <- as.data.frame(lapply(intensity, as.numeric))
rownames(intensity_mat) <- annotation$uniprot_id
```

```r
# Annotation df
annot_df <- as.data.frame(annotation)
rownames(annot_df) <- annotation$uniprot_id

# Metadata df - rownames must match colnames of data
meta_df <- as.data.frame(metadata)
rownames(meta_df) <- metadata$Col_ID

# Assemble
dal <- DAList(
  data       = intensity_mat,
  annotation = annot_df,
  metadata   = meta_df
)

cat(sprintf("DAList assembled: %d proteins x %d samples\n",
            nrow(dal$data), ncol(dal$data)))
```

```
DAList assembled: 3172 proteins x 41 samples
```

3 — Quality Filtering

Protein filtering proceeds in two stages: (1) removal of proteins without evidence of skeletal muscle expression in the Human Protein Atlas (Uhlen et al., 2015), and (2) removal of proteins with excessive missing data. The HPA tissue-relevance filter is a simple process of elimination: if a protein has no record of expression in skeletal muscle, it is unlikely to be biologically meaningful in this context and is removed before statistical modeling.

For DIA-MS data, missing values predominantly arise from peptides below the detection limit (missing not at random, MNAR) rather than stochastic sampling (missing completely at random, MCAR) as is more common in DDA data (Lazar et al., 2016). Label-free proteomics datasets commonly exceed 50% missingness across the full matrix (O'Brien et al., 2018), making per-group proportion filters a practical necessity. Benchmarking by Arioli et al. (2021) using OptiMissP showed that a 50% threshold offered the best trade-off between coverage and data completeness for DIA-MS. McGurk et al. (2020) further demonstrated that missing values in DIA-MS carry biological signal, supporting retention at a moderate threshold. Dabke et al. (2021) compared 80%, 50%, and 30% NA filters and found that 50% retained the most proteins while enabling accurate downstream imputation. Kong et al. (2022) reviewed missingness handling broadly and recommended per-group proportion filters near 50% as a practical default, while Harris et al. (2023) showed that a 50% completeness threshold preserves differential expression detection accuracy across imputation methods. Following these recommendations, we apply a 50% completeness threshold per `Group_Time` condition, ensuring retained proteins have sufficient observations for reliable statistical inference while preserving condition-specific proteins.

DIA-MS search engines export zeros for peptides below the detection limit. These must be recoded as `NA` so downstream functions correctly distinguish absent from zero-abundance measurements.

```r
# Track protein counts through filtering
filter_log <- tibble(
  step      = character(),
  n_before  = integer(),
  n_after   = integer(),
  n_removed = integer()
)

n_start <- nrow(dal$data)

# Convert 0s to NA (DIA-MS exports may use 0 for below-detection-limit)
dal <- zero_to_missing(dal)

n_zeros <- n_start  # same number of proteins, but 0s are now NA
cat(sprintf("Converted zeros to NA. Total proteins: %d\n", nrow(dal$data)))
```

```
Converted zeros to NA. Total proteins: 3172
```

Proteins absent from the HPA skeletal muscle gene list are removed. This tissue-relevance filter ensures downstream analysis focuses on biologically plausible targets rather than environmental or processing artifacts.

```r
n_before <- nrow(dal$data)
dal <- filter_proteins_by_annotation(dal, in_hpa_muscle)
n_after <- nrow(dal$data)

filter_log <- bind_rows(filter_log, tibble(
  step = "HPA muscle tissue filter",
  n_before = n_before, n_after = n_after, n_removed = n_before - n_after
))

cat(sprintf("HPA muscle filtering: %d -> %d proteins (%d removed)\n",
            n_before, n_after, n_before - n_after))
```

```
HPA muscle filtering: 3172 -> 2962 proteins (210 removed)
```

Apply the 50% per-group completeness threshold. The per-group approach preserves condition-specific proteins that a global filter would discard (Webb-Robertson et al., 2015; Kong et al., 2022).

```r
n_before <- nrow(dal$data)
dal <- filter_proteins_by_proportion(
  dal,
  min_prop = 0.50,
  grouping_column = "Group_Time"
)
n_after <- nrow(dal$data)

filter_log <- bind_rows(filter_log, tibble(
  step = "Missingness filter (50% per Group_Time)",
  n_before = n_before, n_after = n_after, n_removed = n_before - n_after
))

cat(sprintf("Missingness filtering: %d -> %d proteins (%d removed)\n",
            n_before, n_after, n_before - n_after))
```

```
Missingness filtering: 2962 -> 1829 proteins (1133 removed)
```

Record protein counts at each stage and write removed proteins with their removal reason (not in HPA muscle vs. missingness) for audit purposes.

```r
# Add the starting count
filter_log <- bind_rows(
  tibble(step = "Raw input", n_before = NA_integer_,
         n_after = n_start, n_removed = NA_integer_),
  filter_log
) %>%
  mutate(pct_retained = round(n_after / n_start * 100, 1))

# Save filtering effects
write_csv(filter_log, file.path(data_dir, "03_filtering_effects.csv"))

# Build filtered proteins list
# Identify which proteins were removed and why
all_uniprots <- annot_df$uniprot_id
```

```r
kept_uniprots <- rownames(dal$data)
removed_uniprots <- setdiff(all_uniprots, kept_uniprots)

filtered_proteins <- annot_df %>%
  filter(uniprot_id %in% removed_uniprots) %>%
  select(uniprot_id, gene, description, in_hpa_muscle) %>%
  mutate(reason = if_else(!in_hpa_muscle, "Not in HPA muscle", "Missingness"))

write_csv(filtered_proteins, file.path(data_dir, "03_filtered_proteins.csv"))

kable(filter_log, caption = "Protein filtering summary")
```

Table 2: Protein filtering summary

step	n_before	n_after	n_removed	pct_retained
Raw input	NA	3172	NA	100.0
HPA muscle tissue filter	3172	2962	210	93.4
Missingness filter (50% per Group_Time)	2962	1829	1133	57.7

A bar chart visualizes protein retention at each step, saved as a PDF for inclusion in downstream consolidated figures.

```r
filter_plot_data <- filter_log %>%
  mutate(step = factor(step, levels = step))

p_filter <- ggplot(filter_plot_data, aes(x = step, y = n_after)) +
  geom_col(fill = "#2166AC", width = 0.6) +
  geom_text(aes(label = n_after), vjust = -0.3, size = 4) +
  labs(x = NULL, y = "Number of proteins",
       title = "Protein retention through filtering pipeline") +
  theme_minimal(base_size = 13) +
  theme(axis.text.x = element_text(angle = 25, hjust = 1))

p_filter
```

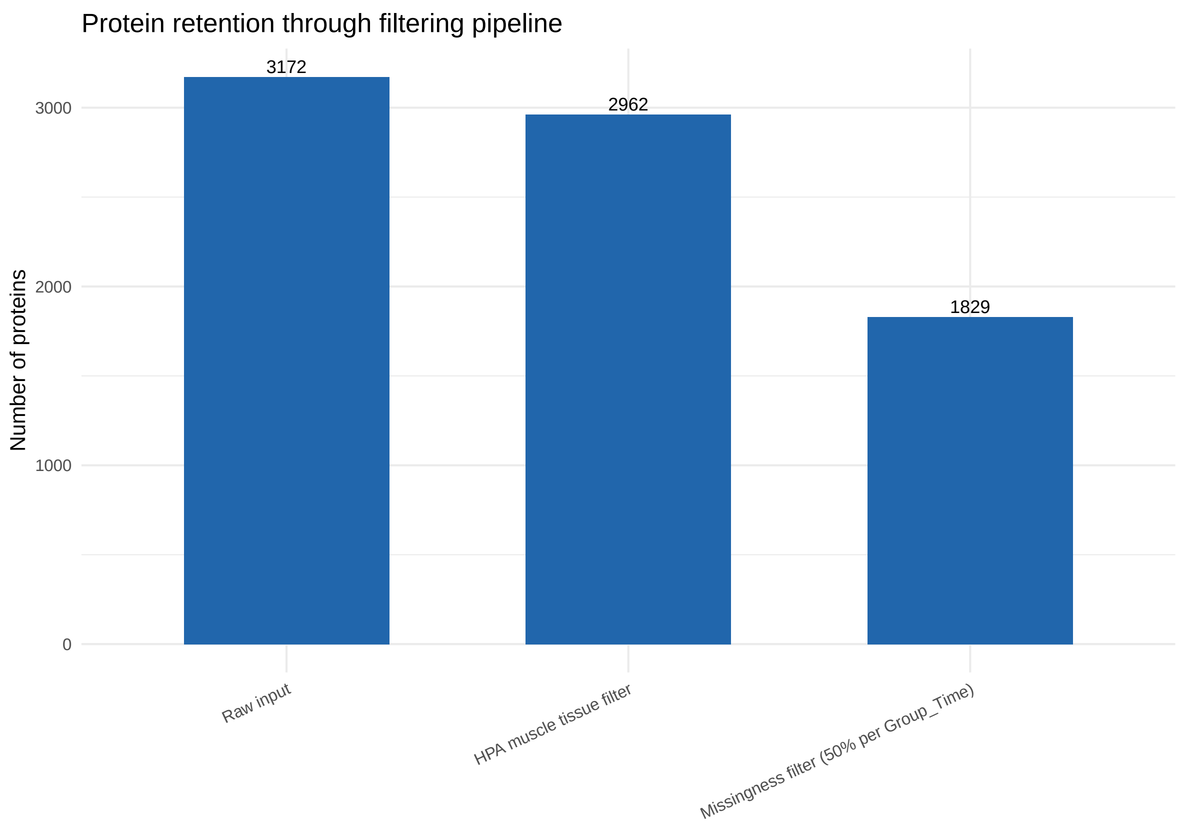

Figure 1: Protein counts at each filtering stage

4 — Outlier Detection (Tiered Approach)

Sample-level outliers can arise from technical failures (failed digestion, poor injection), sample degradation, or mislabeling. We apply three complementary diagnostic methods to flag problematic samples, then use a **tiered strategy** to decide which samples to remove versus retain with downweighting.

Detection methods

1. **Missingness rate** — samples with missing-data percentage exceeding Q3 + 1.5 × IQR (Tukey fence), optionally combined with paired T1/T2 missingness deltas for CR subjects
2. **PCA Mahalanobis distance** — multivariate outlier detection on PC1–3, flagged at ^{2}(0.99, df = 3) (Hubert et al., 2005)
3. **MAD-based median intensity** — samples whose median log2 intensity deviates from the global median by $> 3 \times$ MAD (Leys et al., 2013)

Tiered removal strategy

Rather than removing all samples flagged by 2 methods, we adopt a **graduated approach** based on the limma literature:

- **Catastrophic** (3 flags AND >50% missing): Hard-remove. These represent failed runs where the majority of proteins are undetected; no statistical method can rescue meaningful signal from a sample missing 60%+ of its data.
- **Soft outlier** (2 flags, not catastrophic): Retain in the dataset. These are handled downstream by `arrayWeights()` (Ritchie et al., 2006), which assigns data-driven quality weights to each sample. A marginal sample receives a reduced weight proportional to its deviation, rather than being discarded entirely.
- **Normal** (<2 flags): Full weight.

This strategy is motivated by Gordon Smyth's (limma developer) guidance: *"Arrays have to be very bad indeed before they contain no useful information. It is usually preferable to allow for some samples being less reliable than others in a graduated way rather than complete removal"* (Bioconductor support, Ritchie et al. 2006). The `arrayWeights()` function estimates a log-linear variance model log($^{2}_{gj}$) = $_{g}$ + $_{j}$ across all proteins, where $_{j}$ captures sample-specific quality. Samples with consistently large residuals receive lower weights w_j = 1/exp($_{j}$), achieving data-driven downweighting without arbitrary thresholds (Ritchie et al., 2006; Liu et al., 2015).

For the DEP analysis (step 04), we additionally apply `eBayes(robust = TRUE)` (Phipson et al., 2016), which protects against hypervariable proteins distorting the empirical Bayes prior — a complementary layer of robustness operating at the protein level rather than the sample level.

These diagnostics are applied **before normalization** since they rely on missingness patterns and raw intensity structure rather than normalized values.

Flag samples with anomalously high missingness using IQR fences. For paired CR subjects, a large delta between T1 and T2 missingness may indicate a failed run at one timepoint.

```r
# Per-sample percent missing
pct_missing <- colMeans(is.na(dal$data)) * 100

# Paired missingness for CR subjects with both T1 and T2
paired_subjects <- dal$metadata %>%
  filter(Group != "PPS") %>%
  count(Subject_ID) %>%
  filter(n == 2) %>%
  pull(Subject_ID)

# Per-SUBJECT paired data: one row per subject with max missingness and |delta|
paired_data <- tibble(
  Subject_ID = character(),
  col_t1 = character(),
  col_t2 = character(),
  max_miss = numeric(),
  abs_delta = numeric()
)

for (subj in paired_subjects) {
  rows <- dal$metadata %>% filter(Subject_ID == subj)
  t1_id <- rows$Col_ID[rows$Timepoint == "T1"]
  t2_id <- rows$Col_ID[rows$Timepoint == "T2"]
  if (length(t1_id) == 1 && length(t2_id) == 1) {
    m1 <- pct_missing[t1_id]
    m2 <- pct_missing[t2_id]
    paired_data <- bind_rows(paired_data, tibble(
      Subject_ID = subj,
      col_t1 = t1_id,
      col_t2 = t2_id,
      max_miss = max(m1, m2),
      abs_delta = abs(m2 - m1)
    ))
  }
}

# Map subject-level abs_delta back to per-sample rows for flagging
sample_miss <- tibble(
  Col_ID = colnames(dal$data),
  pct_missing = pct_missing[colnames(dal$data)]
) %>%
  left_join(
    dal$metadata %>% select(Col_ID, Subject_ID),
    by = "Col_ID"
  ) %>%
  left_join(
    paired_data %>% select(Subject_ID, abs_delta),
    by = "Subject_ID"
  )

# IQR-based thresholds
miss_q3  <- quantile(pct_missing, 0.75)
miss_iqr <- IQR(pct_missing)
```

```r
miss_threshold <- miss_q3 + 1.5 * miss_iqr

abs_delta_vals <- paired_data$abs_delta
if (length(abs_delta_vals) > 2) {
  delta_threshold <- quantile(abs_delta_vals, 0.75) + 1.5 * IQR(abs_delta_vals)
} else {
  delta_threshold <- Inf
}

sample_miss <- sample_miss %>%
  mutate(
    miss_flag = pct_missing > miss_threshold |
      (!is.na(abs_delta) & abs_delta > delta_threshold)
  )
```

Multivariate outlier detection in PCA space catches samples that deviate in ways no single metric would flag (Hubert et al., 2005).

```r
# PCA on log2-transformed, median-imputed data
data_for_pca <- dal$data
# Impute NAs with column medians (temporary - for PCA only)
for (j in seq_len(ncol(data_for_pca))) {
  nas <- is.na(data_for_pca[, j])
  if (any(nas)) {
    data_for_pca[nas, j] <- median(data_for_pca[, j], na.rm = TRUE)
  }
}
# No +1 pseudocount needed: zeros already converted to NA by zero_to_missing(),
# and NAs filled by column medians above. All values are positive observed intensities.
data_log2 <- log2(data_for_pca)

pca_res <- prcomp(t(data_log2), center = TRUE, scale. = TRUE)
pc_scores <- pca_res$x[, 1:3]

# Mahalanobis distance
center <- colMeans(pc_scores)
cov_mat <- cov(pc_scores)
mahal_dist <- mahalanobis(pc_scores, center, cov_mat)

# Chi-squared threshold (p < 0.01, df = 3)
mahal_threshold <- qchisq(0.99, df = 3)
```

```
pca_flags <- tibble(
  Col_ID = colnames(dal$data),
  mahal_dist = mahal_dist,
  pca_flag = mahal_dist > mahal_threshold
)
```

MAD-based intensity screening catches global intensity shifts (e.g., loading errors) that PCA may miss (Leys et al., 2013).

```
# Per-sample median log2 intensity
sample_medians <- apply(log2(dal$data), 2, median, na.rm = TRUE)
global_median  <- median(sample_medians)
mad_val        <- mad(sample_medians)

mad_flags <- tibble(
  Col_ID = names(sample_medians),
  sample_median = sample_medians,
  mad_deviation = abs(sample_medians - global_median),
  mad_flag = abs(sample_medians - global_median) > 3 * mad_val
)
```

The three binary flags are combined into a **tiered consensus**. Rather than removing all samples with 2 flags, we classify each sample into one of three tiers using `case_when()`:

- **catastrophic**: 3 flags AND >50% missingness — unambiguous technical failure, hard-removed
- **soft_outlier**: 2 flags but not catastrophic — retained, handled by `arrayWeights()` downstream (Ritchie et al., 2006)
- **normal**: <2 flags — no action needed

This tiered approach preserves marginal samples that still contain useful biological signal while removing only those that are truly unrescuable. The `arrayWeights()` function in the DEP step will assign data-driven quality weights to each sample, effectively downweighting soft outliers in proportion to their deviation without discarding their information entirely.

```
# Combine all flags into tiered classification
outlier_diag <- sample_miss %>%
  select(Col_ID, pct_missing, abs_delta, miss_flag) %>%
  left_join(pca_flags, by = "Col_ID") %>%
  left_join(mad_flags, by = "Col_ID") %>%
  mutate(
    n_flags = miss_flag + pca_flag + mad_flag,
    tier = case_when(
      n_flags >= 3 & pct_missing > 50 ~ "catastrophic",
      n_flags >= 2                     ~ "soft_outlier",
      TRUE                             ~ "normal"
    )
  )

write_csv(outlier_diag, file.path(data_dir, "04_outlier_diagnostics.csv"))

n_catastrophic  <- sum(outlier_diag$tier == "catastrophic")
n_soft          <- sum(outlier_diag$tier == "soft_outlier")
cat(sprintf("Tiered outlier results: %d catastrophic (remove), %d soft (arrayWeights)\n",
            n_catastrophic, n_soft))
```

```
Tiered outlier results: 1 catastrophic (remove), 2 soft (arrayWeights)
```

```
if (n_catastrophic + n_soft > 0) {
  outlier_diag %>%
    filter(tier != "normal") %>%
    select(Col_ID, pct_missing, abs_delta, mahal_dist, n_flags, tier) %>%
    kable(caption = "Flagged samples: catastrophic = remove, soft_outlier = retain with array
}
```

Table 3: Flagged samples: catastrophic = remove, soft_outlier = retain with arrayWeights

Col_ID	pct_missing	abs_delta	mahal_dist	n_flags	tier
CR10_T1	26.79060	25.31438	15.038669	3	soft_outlier
CR386_T1	20.33898	16.56643	7.005708	2	soft_outlier
CR9_T1	60.08748	49.91799	32.895561	3	catastrophic

Three diagnostic panels visualize each outlier method, colored by tier: **red** = catastrophic (hard-removed), **orange** = soft outlier (retained for `arrayWeights`), **gray** = normal. Dashed lines show detection thresholds for each method. (A) paired missingness — one point per subject, x = max missingness of the pair, y = |delta| — with IQR fence, (B) PCA biplot with Mahalanobis threshold, (C) per-sample median intensity with ±3 MAD bounds. All points are labeled with sample/subject identifiers.

```
tier_colors <- c(catastrophic = "#B2182B", soft_outlier = "#F4A582", normal = "gray50")

# Panel 1: Paired missingness - one point per SUBJECT
```

```r
# Determine subject-level tier (worst tier of the pair)
subject_tier <- outlier_diag %>%
  left_join(dal$metadata %>% select(Col_ID, Subject_ID), by = "Col_ID") %>%
  mutate(tier_rank = case_when(
    tier == "catastrophic" ~ 3L,
    tier == "soft_outlier" ~ 2L,
    TRUE ~ 1L
  )) %>%
  group_by(Subject_ID) %>%
  summarise(tier_rank = max(tier_rank), .groups = "drop") %>%
  mutate(tier = case_when(
    tier_rank == 3 ~ "catastrophic",
    tier_rank == 2 ~ "soft_outlier",
    TRUE ~ "normal"
  ))

paired_plot_data <- paired_data %>%
  left_join(subject_tier %>% select(Subject_ID, tier), by = "Subject_ID")

p1 <- ggplot(paired_plot_data, aes(x = max_miss, y = abs_delta)) +
  geom_point(aes(color = tier), size = 3) +
  geom_text_repel(aes(label = Subject_ID, color = tier),
                  size = 2.8, max.overlaps = 30, show.legend = FALSE) +
  geom_hline(yintercept = delta_threshold,
             linetype = "dashed", color = "red", alpha = 0.5) +
  geom_vline(xintercept = miss_threshold,
             linetype = "dashed", color = "red", alpha = 0.5) +
  scale_color_manual(values = tier_colors) +
  labs(x = "Max % Missing (worst of T1/T2)",
       y = "|Delta Missing| (|T2 - T1|)",
       title = "A: Paired Missingness (one point per subject)",
       color = "Tier") +
  theme_minimal(base_size = 11)

# Panel 2: PCA with Mahalanobis - labeled with Col_ID
pc_df <- as.data.frame(pca_res$x[, 1:2])
pc_df$Col_ID <- rownames(pc_df)
pc_df <- left_join(pc_df, outlier_diag %>% select(Col_ID, tier), by = "Col_ID")
var_explained <- round(summary(pca_res)$importance[2, 1:2] * 100, 1)

p2 <- ggplot(pc_df, aes(x = PC1, y = PC2)) +
  geom_point(aes(color = tier), size = 3) +
  geom_text_repel(aes(label = Col_ID, color = tier),
                  size = 2.3, max.overlaps = 50, show.legend = FALSE) +
  scale_color_manual(values = tier_colors) +
  labs(x = sprintf("PC1 (%.1f%%)", var_explained[1]),
       y = sprintf("PC2 (%.1f%%)", var_explained[2]),
       title = "B: PCA Outliers (Mahalanobis)", color = "Tier") +
  theme_minimal(base_size = 11)

# Panel 3: MAD - Col_ID on x-axis + labeled points
p3 <- ggplot(outlier_diag, aes(x = reorder(Col_ID, sample_median), y = sample_median)) +
  geom_point(aes(color = tier), size = 2.5) +
  geom_hline(yintercept = global_median, color = "black") +
  geom_hline(yintercept = global_median + c(-3, 3) * mad_val,
             linetype = "dashed", color = "red", alpha = 0.5) +
  scale_color_manual(values = tier_colors) +
  labs(x = "Sample (ordered by median intensity)", y = "Median log2 intensity",
       title = "C: MAD Intensity Outliers", color = "Tier") +
  theme_minimal(base_size = 11) +
  theme(axis.text.x = element_text(angle = 90, hjust = 1, size = 6))

p_outlier <- p1 + p2 + p3 + plot_layout(ncol = 1, guides = "collect") &
  theme(legend.position = "bottom")
p_outlier
```

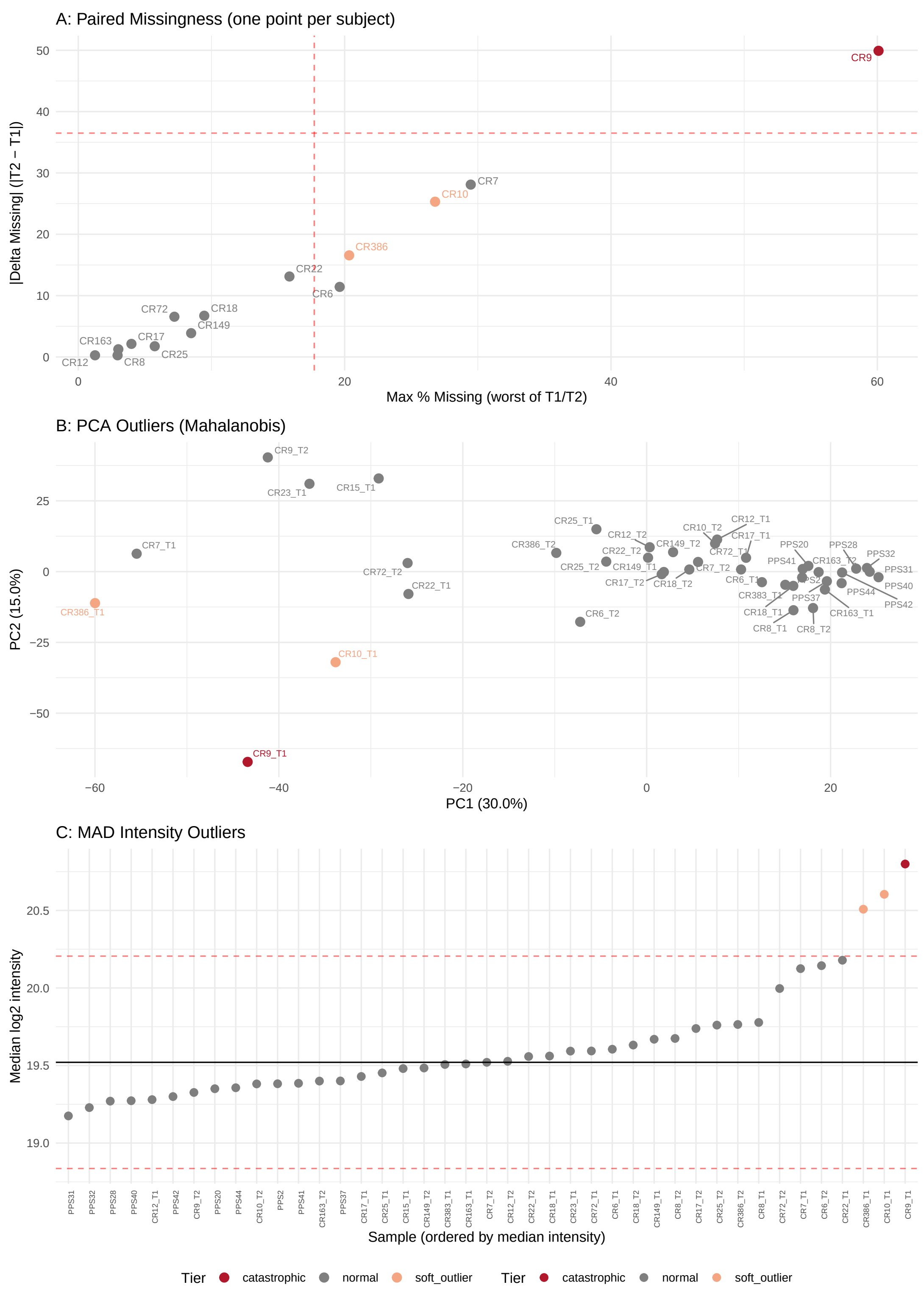

Figure 2: Outlier diagnostic panels — colored by tier (red = catastrophic, orange = soft outlier)

Sample removal

Only **catastrophic** samples are hard-removed via `filter_samples()`. Soft outliers are retained — their reduced quality will be accounted for by `arrayWeights()` (Ritchie et al., 2006) during the DEP analysis in step 04.

`arrayWeights()` works by fitting a log-linear variance model across all proteins, estimating a sample-specific quality factor ${}_j$. Samples with consistently large residuals receive lower weights $w_j = 1/\exp({}_j)$. This data-driven approach is superior to binary removal because:

1. It preserves sample size and degrees of freedom
2. The weight is proportional to actual data quality, not a threshold
3. It avoids creating unbalanced paired designs (orphaned T2 timepoints)
4. `duplicateCorrelation()` handles unbalanced designs correctly (Smyth, Bioconductor support p/125328): *"It reduces the influence of subjects with more biopsies vs those with fewer"*

For paired designs where one timepoint is a soft outlier, `arrayWeights` assigns the problematic timepoint a reduced weight while the good timepoint receives a normal weight. The subject still contributes to the analysis proportionally to its data quality (Liu et al., 2015).

```r
if (n_catastrophic > 0) {
  catastrophic_ids <- outlier_diag %>%
    filter(tier == "catastrophic") %>%
    pull(Col_ID)

  cat(sprintf("Hard-removing %d catastrophic sample(s): %s\n",
              length(catastrophic_ids), paste(catastrophic_ids, collapse = ", ")))

  dal <- filter_samples(dal, !(Col_ID %in% catastrophic_ids))

  cat(sprintf("After removal: %d samples remain\n", ncol(dal$data)))
} else {
  cat("No catastrophic outliers - no samples removed.\n")
}
```

```
Hard-removing 1 catastrophic sample(s): CR9_T1
```

```
After removal: 40 samples remain
```

```r
if (n_soft > 0) {
  soft_ids <- outlier_diag %>% filter(tier == "soft_outlier") %>% pull(Col_ID)
  cat(sprintf("Soft outlier(s) retained for arrayWeights: %s\n",
              paste(soft_ids, collapse = ", ")))
}
```

```
Soft outlier(s) retained for arrayWeights: CR10_T1, CR386_T1
```

5 — Normalization

Normalization corrects for systematic technical variation (e.g., sample loading differences, LC-MS run effects) while preserving biological differences. We use the `proteoDA` normalization report to visually compare eight available methods, then apply cyclic loess (`cycloess`) normalization.

Cyclic loess was recommended by Välikangas et al. (2018) as one of the best-performing methods for label-free quantitative proteomics data, particularly with small sample sizes. It applies a series of pairwise loess regressions between all sample pairs, iteratively adjusting intensity distributions without assuming equal protein loading across conditions — an important property for tissues with potentially different total protein composition (e.g., cancer survivor vs. healthy muscle).

Review the normalization report to confirm cycloess is appropriate before applying it. The report compares eight methods side by side.

```r
write_norm_report(
  dal,
  grouping_column = "Group_Time",
  output_dir = report_dir,
  filename = "01_normalization_report.pdf",
  overwrite = TRUE
)

cat("Normalization report saved to b_reports/01_normalization_report.pdf\n")
```

```
Normalization report saved to b_reports/01_normalization_report.pdf
```

```r
cat("Review this report to confirm cycloess is appropriate for this dataset.\n")
```

```
Review this report to confirm cycloess is appropriate for this dataset.
```

Generate a pre-normalization QC report to serve as the baseline for comparison with the post-normalization QC report. Differences between the two reveal the effect of cycloess correction on sample clustering, correlation structure, and intensity distributions.

```r
write_qc_report(
  dal,
  color_column = "Group_Time",
  label_column = "Col_ID",
  output_dir = report_dir,
  filename = "02_qc_report_pre_norm.pdf",
  overwrite = TRUE
)

cat("Pre-normalization QC report saved to b_reports/02_qc_report_pre_norm.pdf\n")
```

```
Pre-normalization QC report saved to b_reports/02_qc_report_pre_norm.pdf
```

Apply cycloess normalization. Unlike quantile normalization, cyclic loess does not assume equal total protein loading across conditions — important when comparing cancer survivor and healthy muscle tissue (Välikangas et al., 2018; Karpievitch et al., 2012).

```r
dal <- normalize_data(dal, norm_method = "cycloess")

cat(sprintf("Normalization complete (cycloess): %d proteins x %d samples\n",
            nrow(dal$data), ncol(dal$data)))
```

```
Normalization complete (cycloess): 1829 proteins x 40 samples
```

6 — Post-Normalization QC

After normalization, we generate a quality control report to assess sample clustering, correlation structure, and the distribution of normalized intensities. The PCA and hierarchical clustering should show samples grouping primarily by biological condition (`Group_Time`) rather than technical factors.

After normalization, samples should cluster by biology (`Group_Time`) rather than by technical factors. The QC report checks this via correlation heatmap, clustering dendrogram, and PCA.

```r
write_qc_report(
  dal,
  color_column = "Group_Time",
  label_column = "Col_ID",
  output_dir = report_dir,
  filename = "03_qc_report_post_norm.pdf",
  overwrite = TRUE
)

cat("Post-normalization QC report saved to b_reports/03_qc_report_post_norm.pdf\n")
```

```
Post-normalization QC report saved to b_reports/03_qc_report_post_norm.pdf
```

A custom PCA supplements the standard QC report, colored by biological group: CR_T1 (all cancer survivors at baseline, pooling CRE and PLA), CR_T2 (post-training), and PPS_T1 (healthy controls). This three-group view reflects the study's primary biological comparison.

```r
# PCA on normalized data
norm_mat <- dal$data
# Impute remaining NAs with column medians for PCA
for (j in seq_len(ncol(norm_mat))) {
  nas <- is.na(norm_mat[, j])
  if (any(nas)) {
    norm_mat[nas, j] <- median(norm_mat[, j], na.rm = TRUE)
  }
}

pca_norm <- prcomp(t(norm_mat), center = TRUE, scale. = TRUE)
pc_df <- as.data.frame(pca_norm$x[, 1:2])
pc_df$Col_ID <- rownames(pc_df)
pc_df <- left_join(pc_df, dal$metadata, by = "Col_ID") %>%
  mutate(BioGroup = case_when(
    Group_Time %in% c("CRE_T1", "PLA_T1") ~ "CR_T1",
    Group_Time %in% c("CRE_T2", "PLA_T2") ~ "CR_T2",
    TRUE ~ "PPS_T1"
  ))

var_exp <- round(summary(pca_norm)$importance[2, 1:2] * 100, 1)

pal_bio <- c(CR_T1 = "#4393C3", CR_T2 = "#2166AC", PPS_T1 = "#B2182B")
```

```
p_pca <- ggplot(pc_df, aes(x = PC1, y = PC2, color = BioGroup)) +
  geom_point(size = 3.5, alpha = 0.85) +
  stat_ellipse(aes(group = BioGroup), type = "norm", level = 0.68,
               linetype = "solid", linewidth = 0.7) +
  geom_text_repel(aes(label = Col_ID), size = 2.3, max.overlaps = 50,
                  show.legend = FALSE) +
  scale_color_manual(values = pal_bio) +
  labs(x = sprintf("PC1 (%.1f%%)", var_exp[1]),
       y = sprintf("PC2 (%.1f%%)", var_exp[2]),
       title = "Post-normalization PCA",
       color = "Group") +
  theme_minimal(base_size = 12) +
  theme(legend.position = "bottom")

p_pca
```

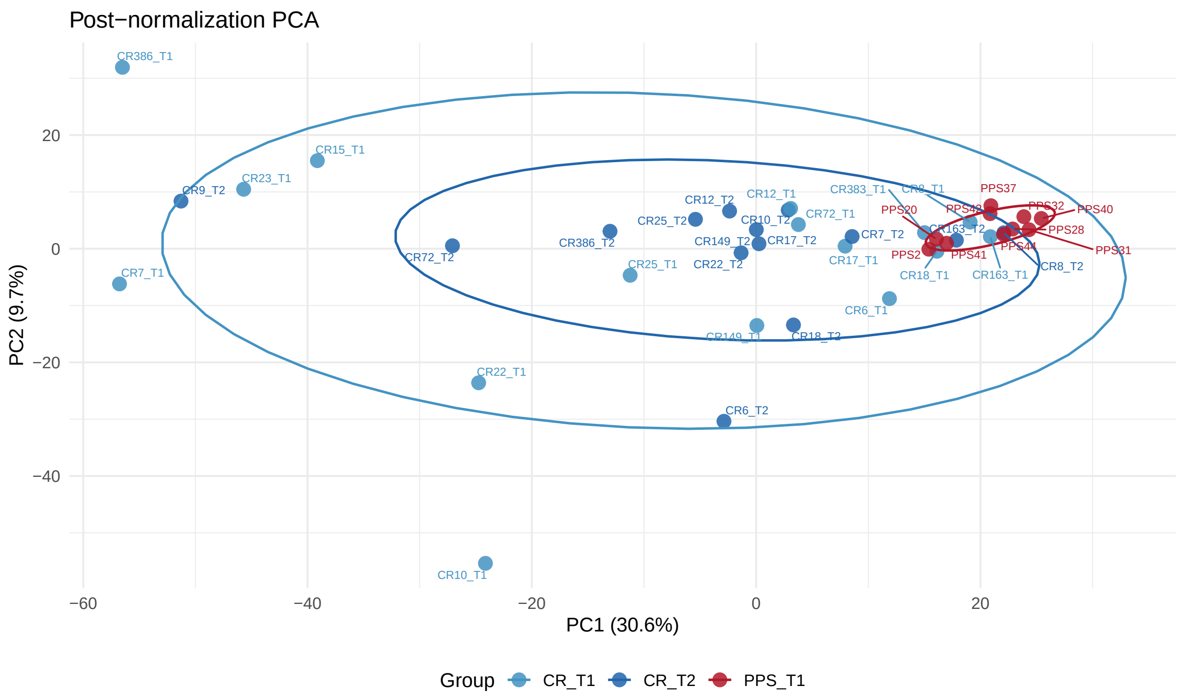

Figure 3: Post-normalization PCA — CvH dataset (CR_T1 / CR_T2 / PPS_T1)

7 — Export

We export the normalized dataset in two formats: a flat CSV for interoperability with other tools, and an `.rds` file preserving the full `DAList` object for seamless continuation into the modeling phase.

Export both a flat CSV (for interoperability with external tools) and the full DAList object (for seamless continuation into downstream stages).

```r
# CSV: annotation (including n_seq for downstream peptide counts) + normalized intensities
export_df <- bind_cols(
  as_tibble(dal$annotation) %>%
    select(uniprot_id, protein, gene, description, n_seq),
  as_tibble(dal$data)
)

write_csv(export_df, file.path(data_dir, "01_normalized.csv"))

# RDS: full DAList
saveRDS(dal, file.path(data_dir, "01_DAList_normalized.rds"))

cat(sprintf("Exported: %d proteins x %d samples\n",
            nrow(dal$data), ncol(dal$data)))
```

```
Exported: 1829 proteins x 40 samples
```

```r
cat(sprintf("  CSV: %s\n", file.path(data_dir, "01_normalized.csv")))
```

```
  CSV: /Users/dtl0018/Desktop/A_Proteomics_Analysis/A_CvH_2026/01_normalization/c_data/01_no
```

```r
cat(sprintf("  RDS: %s\n", file.path(data_dir, "01_DAList_normalized.rds")))
```

```
  RDS: /Users/dtl0018/Desktop/A_Proteomics_Analysis/A_CvH_2026/01_normalization/c_data/01_DA
```

8 — Downstream Preview

The normalized `DAList` feeds into three subsequent stages, each with its own script and narrative QMD:

- **02_Imputation**: MAR/MNAR classification, benchmark of 14 methods, quality audit

- **03_Sensitivity_check**: 12-pipeline grid (3 norms $\times$ 4 imps) confirming robustness
- **04_DEP**: Differential expression via limma with arrayWeights, duplicateCorrelation, and eBayes(robust=TRUE)

References

1. Välikangas T, Suomi T, Elo LL (2018). A systematic evaluation of normalization methods in quantitative label-free proteomics. *Briefings in Bioinformatics* 19(1):1-11.
2. Ritchie ME, Phipson B, Wu D, et al. (2015). limma powers differential expression analyses for RNA-sequencing and microarray studies. *Nucleic Acids Research* 43(7):e47.
3. Lazar C, Gatto L, Ferro M, et al. (2016). Accounting for the multiple natures of missing values in label-free quantitative proteomics data sets to compare imputation strategies. *Journal of Proteome Research* 15(4):1116-1125.
4. Webb-Robertson BM, Wiber HK, Matzke MM, et al. (2015). Review, evaluation, and discussion of the challenges of missing value imputation for mass spectrometry-based label-free global proteomics. *Journal of Proteome Research* 14(3):920-930.
5. Herbrich SM, Cole RN, West KP Jr, et al. (2013). Statistical inference from multiple iTRAQ experiments without using common reference standards. *Journal of Proteome Research* 12(2):594-604.
6. Phipson B, Lee S, Oshlack A, et al. (2016). Robust hyperparameter estimation protects against hypervariable genes and improves power to detect differential expression. *Annals of Applied Statistics* 10(2):946-963.
7. Hubert M, Rousseeuw PJ, Vanden Branden K (2005). ROBPCA: A new approach to robust principal component analysis. *Technometrics* 47(1):64-79.
8. Leys C, Ley C, Klein O, et al. (2013). Detecting outliers: Do not use standard deviation around the mean, use absolute deviation around the median. *Journal of Experimental Social Psychology* 49(4):764-766.
9. Karpievitch YV, Dabney AR, Smith RD (2012). Normalization and missing value imputation for label-free LC-MS analysis. *BMC Bioinformatics* 13(Suppl 16):S5.
10. O'Brien JJ, Bhatt DK, Engel JM, et al. (2018). Label-free quantitative proteomics analysis of mass spectrometry data with implications for missingness and filtering strategies. *Annals of Applied Statistics* 12(4):2484-2510. PMID:30473739.
11. Arioli A, Dagliati A, Geary B, et al. (2021). OptiMissP: A dashboard to assess missingness in proteomic data-independent acquisition mass spectrometry. *PLoS One* 16(4):e0249771. PMID:33857200.

12. McGurk KA, Dagliati A, Chiasserini D, et al. (2020). The use of missing values in proteomic data-independent acquisition mass spectrometry to enable disease activity discrimination. *Bioinformatics* 36(7):2217-2223. PMID:31790148.

13. Dabke K, Hendrick G, Devkota S (2021). The gut microbiome and metabolic syndrome — proteomics quality-filtering benchmarks. *Journal of Proteome Research* 20(6):3214-3229. PMID:33939434.

14. Kong W, Hui HWH, Peng H, et al. (2022). Dealing with missing values in proteomics data. *Proteomics* 23(23-24):e2200092. PMID:36349819.

15. Harris L, Fondrie WE, Oh S, Noble WS (2023). Evaluating proteomics imputation methods with improved criteria. *Journal of Proteome Research* 22(11):3427-3438. PMID:37861703.

16. Uhlen M, Fagerberg L, Hallstrom BM, et al. (2015). Tissue-based map of the human proteome. *Science* 347(6220):1260419. PMID:25613900.

17. Ritchie ME, Diyagama D, Neilson J, et al. (2006). Empirical array quality weights in the analysis of microarray data. *BMC Bioinformatics* 7:261. PMID:16712727.

18. Liu R, Holik AZ, Su S, et al. (2015). Why weight? Modelling sample and observational level variability improves power in RNA-seq analyses. *Nucleic Acids Research* 43(15):e97. PMID:25925576.

Session Info

```r
sessionInfo()
```

```
R version 4.5.1 (2025-06-13)
Platform: aarch64-apple-darwin20
Running under: macOS Tahoe 26.2

Matrix products: default
BLAS:   /Library/Frameworks/R.framework/Versions/4.5-arm64/Resources/lib/libRblas.0.dylib
LAPACK: /Library/Frameworks/R.framework/Versions/4.5-arm64/Resources/lib/libRlapack.dylib;  

locale:
[1] en_US.UTF-8/en_US.UTF-8/en_US.UTF-8/C/en_US.UTF-8/en_US.UTF-8

time zone: America/Chicago
tzcode source: internal
```

```
attached base packages:
[1] stats     graphics  grDevices utils     datasets  methods   base

other attached packages:
 [1] here_1.0.2      knitr_1.51      ggrepel_0.9.6   patchwork_1.3.2
 [5] ggplot2_4.0.2   stringr_1.6.0   tidyr_1.3.2     dplyr_1.2.0
 [9] readr_2.1.6     readxl_1.4.5    proteoDA_1.0.1

loaded via a namespace (and not attached):
 [1] gridExtra_2.3        rlang_1.1.7          magrittr_2.0.4
 [4] clue_0.3-66          GetoptLong_1.1.0     otel_0.2.0
 [7] matrixStats_1.5.0    compiler_4.5.1       mgcv_1.9-4
[10] png_0.1-8            systemfonts_1.3.1    vctrs_0.7.1
[13] shape_1.4.6.1        pkgconfig_2.0.3      crayon_1.5.3
[16] fastmap_1.2.0        magick_2.9.0         labeling_0.4.3
[19] rmarkdown_2.30       tzdb_0.5.0           preprocessCore_1.72.0
[22] ragg_1.5.0           purrr_1.2.1          bit_4.6.0
[25] xfun_0.56            aplot_0.2.9          jsonlite_2.0.0
[28] cluster_2.1.8.2      parallel_4.5.1       R6_2.6.1
[31] stringi_1.8.7        RColorBrewer_1.1-3   limma_3.66.0
[34] cellranger_1.1.0     Rcpp_1.1.1           iterators_1.0.14
[37] pacman_0.5.1         IRanges_2.44.0       Matrix_1.7-4
[40] splines_4.5.1        tidyselect_1.2.1     yaml_2.3.12
[43] doParallel_1.0.17    codetools_0.2-20     affy_1.88.0
[46] lattice_0.22-9       tibble_3.3.1         Biobase_2.70.0
[49] treeio_1.32.0        withr_3.0.2          S7_0.2.1
[52] evaluate_1.0.5       gridGraphics_0.5-1   circlize_0.4.17
[55] pillar_1.11.1        affyio_1.80.0        BiocManager_1.30.27
[58] ggtree_3.16.3        foreach_1.5.2        stats4_4.5.1
[61] ggfun_0.2.0          generics_0.1.4       vroom_1.7.0
[64] rprojroot_2.1.1      hms_1.1.4            S4Vectors_0.48.0
[67] scales_1.4.0         tidytree_0.4.7       glue_1.8.0
[70] lazyeval_0.2.2       tools_4.5.1          vsn_3.78.0
[73] fs_1.6.6             grid_4.5.1           ape_5.8-1
[76] colorspace_2.1-2     nlme_3.1-168         cli_3.6.5
[79] rappdirs_0.3.4       textshaping_1.0.4    ComplexHeatmap_2.24.1
[82] gtable_0.3.6         yulab.utils_0.2.4    digest_0.6.39
[85] BiocGenerics_0.56.0  ggplotify_0.1.3      rjson_0.2.23
[88] farver_2.1.2         htmltools_0.5.9      lifecycle_1.0.5
[91] GlobalOptions_0.1.3  statmod_1.5.1        bit64_4.6.0-1
[94] MASS_7.3-65
```